FERMILAB-Conf-97/314

# The Fermilab DART Data Acquisition System at Running Experiments

Carmenita Moore
For the DART Collaboration
*Fermi National Accelerator Laboratory*
*P.O. Box 500, Batavia, Illinois 60510*

September 1997

Presented at the *1997 Real Time Conference*, Beaune, France, September 22-26, 1997

To appear in *IEEE Transactions on Nuclear Science*

# The Fermilab DART Data Acquisition System at Running Experiments [1]

Carmenita Moore, For the DART collaboration, Online Systems Department
Electronics Systems Engineering Department
E781, E815, E831, KTeV, E835, E871, E872
Fermilab, P.O. Box 500, Batavia, Illinois 60150

*Abstract*

The full DART system described in previous papers [1],[2], has been running in the Fermilab Fixed target program successfully since June 1996, when the first beam was delivered to seven experiments using the system. Since then several test beam experiments have decided to adopt parts or all of DART.

The DART data acquisition system is designed to meet the needs of a number of experiments with a wide range of requirements: from a few hundred KB/s at low rates to one of the highest rate HEP data acquisition systems in operation, 160MB/s, for KTeV. What stands out in DART is its fully distributed nature, its high degree of tailor-ability, and its wide capabilility of readout rates and throughput, while at the same time maintaining high levels of integration.

We report on the status of DART and additions that have been and are being developed for the new customers. We also review experiences gained from use at running experiments and future extensions for use by test beams including D0 and MINOS.

## I. INTRODUCTION

DART is a collaboration between Fermilab experiments and the Fermilab Computing Division to provide a generally usable DA system for the Fixed Target running period 96-97. The run started ramping up in June of 96, and culminated with eight experiments taking data with DART. The experiment data acquisition systems have performed reliably since the start of commissioning. The strategy of incremental versions of DART to support commissioning and then full data taking proved successful and resulted in data acquisition not becoming a critical path item for any of the experiments. As of February, the high rate KTeV experiments has recorded approximately 2.5 billion events onto 20 TB of dlt tapes.

In DART, front-end readout controllers push data over multiple RS-485 "stream" readout cables into programmable dual ported buffer memories residing in VME "event building" crates. A single event spans across streams, and its pieces are directed to a single VME crate. Events are cycled through VME crates in a round-robin fashion, and each VME crate is connected to a SGI Challenge processor via a Performance Technology (PTI) high speed VME-VME link. The Challenge reads the event table and fragments from the buffer memories into its own memory, which is managed by DART software.The highest bandwidth user, KTeV, had 6 streams, 3 event building VME crates, and an online analysis crate, for a total of 24 buffer memory nodes.

DART is a fully distributed system, designed to start up, control, and monitor an arbitrary number of computer nodes, and efficiently handle replicated nodes. It is also highly customizable at all levels. DART provides a mechanism to define and automatically start up multiple DA configurations. Application control is performed in a parallel, but synchronized fashion. DART software also provides

- a distributed information system for configuring the DA, recording run history, and temporary storage of statistical information
- a system for local buffer management and providing services on the buffers
- a data logger for tape and disk files
- an error reporting, logging, and display system
- a system for providing back-end workstations with live data over the Internet and an analysis framework to process the data and display it in PAW or HISTOSCOPE and lastly
- a sophisticated graphical statistics monitoring system for monitoring the entire DA.

## II. DART FIXED TARGET RUN STATUS

DART experiments have recently completed the 96-97 fixed target run. Overall, the DART system remained extremely robust. On-call support was characterized by a minimum of support calls and these were for the most part computer hardware failures rather than software failures.

The DART hardware was stable with the exception of a few ECOs made for adjusting to the realities of experiment environments (e.g. ground loop and transmission line effects). At one experiment fiber optic transmitter ICs were operating below their design temperature due to the Chicago winter air coming through a cooling vent. In addition, problems with extra or missing data words had to be resolved.

After resolving the usual problems of memory leaks and corruption in some DART components during commissioning, (see the Dart Integration Experiences section below), online software also remained both stable and reliable. Software upgrades were limited to minor fixes or enhancements such as new diagnostic tools. Memory leaks and corruption problems of this nature are expected in early releases of software and were fixed during the commissioning phase of the run and not during full data taking.

[1]This work is sponsored by DOE contract No. DE-AC02-76CH03000

## III. Test Beam Experiments

We are currently working on integrating DART into a number of test-beam DAs. Integration with older parts of the CDF Run I software being used for the test-beam DA went quite well.

Integration of DART with EPICs in the D0 test-beam is ongoing. They use all the major components for the Unix, such as the error reporting, configuration management, startup ,etc. Their initial use of the TCP/IP event transport software, p2p, to buffer an event and transfer it over a socket to a UNIX box, was modified to a multi-source scheme for VME. Implementation of a monitoring client capable of handling event data as well as EPICS output is still pending.

The MINOS test-beam was successful in its use of DART. They also employed the major Unix components of DART, and also included use of CAMAC libraries written by E872, for 2249 and 2280 modules, for data readout through a CBD.

## IV. DART Integration Experiences

The strategy of incremental releases of the DART system to the experimenter to support on going DA activities proved invaluable, for both debugging and integration testing among the code developers and the users alike, allowing problems to be uncovered early enough such that no data acquisition system became a critical path leading into the run.

Early releases of some software components had low level memory leaks, segmentation faults, or corruption problems of some form. Occurrences of these problems, ranged from once every few hours to once per month. Because of their low frequency and non-local nature, these bugs are very difficult to track down. In light of this difficulty, we began pursuing the use of software tools and techniques to improve the quality of the online software. A couple of months into the run, we began evaluating Parasoft's Insure++ [5] and PureAtria's Purify [6] tools for analysis of runtime, and memory errors. Both products are used by factions within the Computing Division, but for our purposes, use of Insure++ was more extensive.

Insure++ detects programming and runtime errors, memory corruption and memory leaks, including compile time, runtime and third party errors from other packages such as X, Motif, curses and Unix system calls. It was evaluated on an older version of the configuration management software, dis, known to have leak and corruption problems, which were fixed in subsequent releases after man hours of debugging. Insure++ located all the known problems plus some, including compile time errors. The pinnacle for this tool was solving a once per day crash in the monitoring software, damp, occurring at only one experiment. It took a weekend of running the experiment's damp configuration on our test-stand before the offending code path was traversed. The bug was identified by the tool as a double memory free. We have since "insured" all of our core software and recommend use of Insure++ in C and C++ code development.

Another technique employed in improving software quality was code walkthroughs [3], [4], again motivated by a recurring corruption caused by error in code logic but undetectable by Insure++. The logic error was found and fixed as a result of the walkthrough. Walkthroughs were also used to evaluate the logic of algorithms in other core components with positive results.

## V. Current Work and Future Plans

A Fixed Target Run in 1999 is under consideration by Lab management. In preparation, we are working on extensions to the DART hardware and software systems. Possible changes include the port of DART components to other platforms, investigation of hardware alternatives to increase event throughput from VME to UNIX, and a Java based variant of an error reporting/monitoring system.

### *A. DART Software Modifications*

Much of the recent work on DART has been modifications and/or enhancements to the existing components :

- The operator control panel (ocp) [14] was modified to include a logger "state display", besides the logger statistics fed to the monitoring system for display. The addition of a graphical editor for editing the system configuration parameters, still only in the design phase.
- the TCP/IP event transport software written at E872, p2p, was extended to a multi-source gateway for the D0 test beam.
- extensions to DART include porting some of the components to NT and/or Linux. At this writing, daft has been ported to NT and the IEEE CAMAC driver has been ported to both Linux and NT.

There is general interest in porting DART components to Linux by the Computing Division, and experimenters. Due to the load on the monitoring an slow control nodes of their DA, KTeV, is considering porting to Linux. Since engineers and hardware module developers/testers use PCs the availability of test-stand software for NT is advantageous.

### *B. Alerts*

New on the scene is a CDF effort along with the Fermilab Computing Division to develop a prototype error reporting/monitoring subsystem for the CDF Run II Vertical Slice Test, [10]. Its goal is to use NDDS (Network Data Delivery Service) [7], to transfer error and statistical data from VME to UNIX for display and monitoring in Java applets, which dynamically update, running as an application or in a browser. NDDS employs a dissemination architecture, where producers publish data into "the network" and consumers subscribe to data from "the network"; neither knows where the data goes or originates.

The alerts error reporting/monitoring system will be incorporated into DART as a variant of the current error reporting (murmur) and monitoring (damp) subsystems.

One of the motivations for alerts is ease of access to error and statistical DA information via generic web interface (hence the use of Java), from whereever a web interface is available.

Advantages of the alerts error reporter over murmur, include more flexible control over routing messages to displays, greater control on message attributes provided by filter pattern matching applied to messages to display glyphs, run scripts, play sounds, etc.(see jems section below) in conjunction with our efforts to provide more sophisticated, user friendly, portable software to our experimenters.

*1) jems*

The error reporting system, jems (Java Error Message System), is composed of an error reporting server, written in Java, which acts as a consumer of NDDS error messages that it in turn transfers over TCP/IP sockets to its Java error clients for display. Error status Messages are sent to an error reporting client based on filters sent to the server from the client. A filter can be a specific string, or a string including wildcards and/or regular expressions. When the server receives an NDDS error message, it performs pattern matching of the message against each of its clients list of filter requests. A message is sent to every client whose filter the message matched.

The design of the jems error display applet allows color display of error messages. By default, color is based on the severity of the message. The message display can be configured dynamically or by an html file to base message color on the nodename, application name or filter pattern matching applied to the text of the message. This second level of filter pattern matching is internal to the applet and is separate from the the filtering the server performs to route messages to the client. The display of an error message, can also include sound or image, based on the filter pattern matching.

A test version of jems has been recently released. Only basic functionality is implemented:

- color display of messages based on their severity
- dynamic configuration to change the background color and size of the display, and the font size of messages
- filter configuration to specify the filter list sent to the jems error server, removal of those requests from the server i.e. the server no longer sends messages matching that filter to the client, and deletion of the filter completely from the applet.

*2) jams*

Investigation and design of the monitoring system, jams (Java Alarms Monitoring System), is underway. Its main components will include a server implemented in Java, designed to handle both NDDS and socket client consumers, and Java applets using widget libraries for data monitoring and display.

A number of third parties have been written class libraries to fill in the gaps of Java's AWT class libraries. Missing are such commonly used widgets as image buttons, tabbed folders, scrolled windows, progress bars etc.

As part of the survey of third party libraries for the error and monitoring system, some useful components and properties have been defined [11], based on the lacks in the standard Java AWT classes and features and functionality considered desirable:

- Scrolled Window. Implementing these with the basic AWT classes is tedious, and it is difficult to obtain adequate performance. A good scrolled window is essential for any serious work. One is included in the basic Java 1.1 widget set. An ideal implementation would support live scrolling.
- Extended Button. The Button class in the AWT does not allow multi-line text, or images. Also, mouse events are not delivered so that the standard "help" pop-up cannot be done. A good button class would allow all these things.
- Tabbed Folders. While the AWT CardLayout allows these to be implemented, this is tedious. A good implementation would allow multiple rows of tabs.
- Multicolumn List.
- Hierarchical Tree. This should support icons as tree labels. Ideally it should also allow multiple columns.
- Progress Bar. This should allow optional display of the percent number.
- Toolbar. This could be trivially implemented with a reasonable Button class so is not crucial.
- Convenience Dialogs. Analogous to the Motif and/or tk question, message etc. boxes. These are sufficiently easy to implement so as to not be crucial.
- Combo box (combined textfield/choice).
- Graphs/charts. Needed in some specialized applications such as jams.

According to the above requirements and jams need for graphs and charts, especially a strip-chart, we are evaluating NetFactory's NetCharts [9] and KL Group's JClass Chart [8].

Definition and design of the interface to handle the communication of dynamic data updates from the NDDS environment into that of the chosen Java widget library for display is also being done.

## C. Hardware

Increased rate of VME event throughput is an issue for E872 and possibly D0. The currently supported links from the "event building" VME crate are the PTI VME to VME, or Ethernet. While the slower (< 400 KB/s) Ethernet option exists, change in the accelerator beam spill cycle for the 1997 run will require greater throughput, requiring the DA computer to have a VME bus, such as an SGI challenge, and this is expensive. We are now exploring use of the BIT3 link, which supports other busses such as gio, to fill this performance gap. This will permit cheaper filter/logging computers, such as an SGI Indy, to be used. Since the BIT3 does not support VME D64, but rates of up to 20MB/s are available.

We are exploring upgrading the DART data cable links from the front end readout crates to the event builder by doing R&D on a new data link protocol and interfaces [15]. The goal is to provide for low cost connection to parallel links of 100

Mbytes/sec from the data sources, over fiber optic or copper to a 64 x 64 cross bar switch and delivery to the level 3 filter processors. The unique features of the design of this system include hardware level system auto-initialization, easy ability to cascade switches in an overall DA architecture, flexibility in the configuration from small to large systems - something that has proven of actual value in DART - and attention to diagnostics and monitoring during the hardware architecture design and implementation - again something that has proven its worth in the current DART hardware implemention.

## VI. Conclusion

DART has been successfully incorporated into eight experiments which comprise a wide range of requirements, rates, and architectures. The goals defined at DART's inception, to provide a general, customizable, and robust DA system to meet the needs of these experiments was successfully achieved.

We are actively researching and evaluating tools and techniques to improve the quality of software, including our flavor of code inspection techniques as they apply to the development of software in the high energy physics community.

We hope that the experiences gained in Fixed Target 96-97 will place us further ahead in our DA system development for Fixed Target 99.

## VII. References

[1] "Extending DART to Meet the Data Acquisition Needs of Future Experiments at Fermilab", Proceedings of CHEP95

[2] www-dart.fnal.gov:8000, DART collaboration home page.

[3] E.Berman "Software Inspections at Fermilab - Use and Experience" RT97 Conference Proceedings

[4] www-ols.fnal.gov/ols/www/process/ftr.html, Formal Technical Reviews Web page

[5] www.parasoft.com/insure, Parasoft's Insure++ Web page

[6] www.pureatria.com, www.pureatria.com, PureAtria, Purify Web page

[7] www.rti.com, Real-Time Innovations, Inc., NDDS Web page

[8] www.klg.com/jclass, KLG Group's JCLASS/JCHART Web page

[9] www.netcharts.com, NetFactory's NetChart Web page

[10] www-ols.fnal.gov/ols/www/alerts/alerts.html, Alerts Project Web page

[11] www-b0.fnal.gov:62000/programming/Java/awtextensions.html, CDF's AWT Extensions Web page.

[12] www.fnal.gov/cd/serial_media/serial-media.html, Serial Media Web page.

[13] www.fnal.gov/fermitools/abstracts/camac/abstract.html, Fermitools CAMAC Web page

[14] L.Mengel, et al. "Operator Control Program (ocp) User's Guide" available from URL, fndaub.fnal.gov:8000/vcurrent/vdart.html

[15] dahserv.fnal.gov/daq_randd/specifications/system_spec_working_draft.htm, DALite Data Acquisition System Web Page